VAX FORTRAN
Installation Guide/Release Notes

Order Number: AA-H953E-TE

digital

software

VAX FORTRAN Installation Guide/Release Notes

Order Number: AA-H953E-TE

August 1985

This document is organized into two parts. The first part (Chapter 1) contains instructions for installing the VAX FORTRAN compiler on a VAX/VMS V4 operating system. The second part (Chapter 2) contains information about a variety of topics related to VAX FORTRAN V4.0 and its update releases.

Revision/Update Information:	This revised document supersedes the *VAX FORTRAN Installation Guide/Release Notes* (Order No. AA-H953D-TE).
Operating System and Version:	VAX/VMS Version 4.0 or later
Software Version:	VAX FORTRAN Version 4.3

digital equipment corporation
maynard, massachusetts

First Printing, September 1984
Revised, August 1985

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem-10	PDP	VT
DECSYSTEM-20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital

ZK-2610

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a registered trademark of the American Mathematical Society.

Contents

PREFACE vii

CHAPTER 1 INSTALLATION GUIDE 1-1

1.1 INSTALLATION PREREQUISITES 1-2
- 1.1.1 Small-Disk System Installation Requirements 1-2

1.2 CONTENTS OF THE SOFTWARE INSTALLATION PACKAGE 1-3

1.3 INSTALLATION PROCEDURE 1-4
- 1.3.1 Installing the VAX FORTRAN Compiler and FORSYSDEF.TLB 1-5
- 1.3.2 Installing the VAX FORTRAN HELP File 1-9

1.4 DEFINING VAX FORTRAN AS A KNOWN SHARED IMAGE WITH THE INSTALL UTILITY 1-11
- 1.4.1 Preparation 1-12
- 1.4.2 Using INSTALL at System Bootstrap Time 1-13
- 1.4.3 Using the INSTALL Utility During Timesharing 1-14

1.5 UPDATE RELEASES 1-14
- 1.5.1 Updates to the Compiler Kit 1-15
  - 1.5.1.1 Updating FORSYSDEF • 1-15
- 1.5.2 Updates to the HELP Kit 1-16

1.6 EDIT HISTORY FILE 1-16

CHAPTER 2 RELEASE NOTES 2-1

2.1 VAX FORTRAN V4 ENHANCEMENTS AND LANGUAGE EXTENSIONS 2-2
2.1.1 VAX FORTRAN V4.0 Notes 2-2
2.1.2 VAX FORTRAN V4.1 Notes 2-4
2.1.3 VAX FORTRAN V4.2 Notes 2-5
2.1.4 VAX FORTRAN V4.3 Notes 2-6

2.2 FORSYSDEF CHANGES 2-6

2.3 KNOWN PROBLEMS WITH THE VAX FORTRAN V4.3 COMPILER 2-7

2.4 VAX FORTRAN RTL CHANGES 2-7
2.4.1 RTL V4.0 Behavior Changes 2-8
2.4.2 RTL V4.1 Fix Descriptions 2-8
2.4.3 RTL V4.2 Fix and Restriction Descriptions 2-9

2.5 COMPATIBILITY ISSUES INVOLVING VAX FORTRAN V3 AND V4 2-9
2.5.1 Source Compatibility 2-9
2.5.2 Object Compatibility 2-9
2.5.3 Image Compatibility 2-10

2.6 RUN-TIME BEHAVIOR 2-10
2.6.1 Flow Boolean Order 2-10
2.6.2 Subtle Bounds Errors 2-11
2.6.3 DMOD More Accurate 2-11
2.6.4 Use of VAX/VMS Real-Time Features 2-11
2.6.5 Unformatted Writes of INTEGER*2 Parameter Constants 2-12

2.7 HELP OPTIONS FOR VAX FORTRAN V4 2-13
2.7.1 Long HELP 2-13
2.7.2 Short HELP 2-13

2.8 COMPILER PERFORMANCE AND RECOMMENDED WORKING SET 2-13
2.8.1 CPU Usage 2-14
2.8.2 Memory Usage 2-14

2.9 OUTPUT FILE SPECIFICATIONS ON COMMAND LINES 2-20
2.9.1 Behavior of VAX FORTRAN V3 2-20
2.9.2 Behavior of VAX FORTRAN V4 2-21
2.9.3 How to Change Command Files 2-21

2.10 USAGE RECOMMENDATIONS 2-22

2.11 INFORMATION ABOUT BUG FIXES 2-22

2.12 DOCUMENTATION ERRORS 2-23
2.12.1 Changes to On-Line HELP Documentation 2-23
2.12.2 Changes to *Programming in VAX FORTRAN* 2-23
2.12.3 Changes to *VAX FORTRAN User's Guide* 2-28
2.12.4 Changes to *VAX FORTRAN Language Summary* 2-31

APPENDIX A SOFTWARE PERFORMANCE REPORTS A-1

FIGURES
2–1 Page Faults Versus Working Set, VAX FORTRAN V3 2-15
2–2 Page Faults Versus Working Set, VAX FORTRAN V4 2-16
A–1 Software Performance Report (SPR) Form A-3

TABLES
2–1 Program Descriptions for Figures 2-1 and 2-2 2-17

Preface

This manual provides a step-by-step procedure for installing the VAX FORTRAN compiler on a VAX/VMS system. In addition, it provides FORTRAN programmers with information on a variety of topics that pertain to VAX FORTRAN V4.0 and its update releases.

Intended Audience

This manual is intended for system managers and FORTRAN programmers with a working knowledge of VAX/VMS.

Structure of This Document

This manual is organized into two chapters and one appendix.

Chapter 1 provides information on the following:

- Installation prerequisites
- Contents of the software installation package
- The procedure for installing the VAX FORTRAN compiler, which builds FORSYSDEF.TLB on site
- The procedure for installing the VAX FORTRAN HELP file
- The procedure for installing the VAX FORTRAN compiler as a known shared image
- Information on maintenance updates

Chapter 2 provides information on the following:

- Changes associated with VAX FORTRAN V4.0 and the update releases V4.1, V4.2, and V4.3 (see Section 2.1)
- Changes to FORSYSDEF (see Section 2.2)
- Known problems with the VAX FORTRAN V4.3 compiler (see Section 2.3)
- Changes implemented in VAX FORTRAN RTL (see Section 2.4)

- Compatibility of VAX FORTRAN V3 and VAX FORTRAN V4 (see Section 2.5)
- Run-time behavior differences between VAX FORTRAN V3 and VAX FORTRAN V4 (see Section 2.6)
- On-line HELP options (see Section 2.7)
- Compiler performance measurements and suggestions for maximizing performance (see Section 2.8)
- Changes to the FORTRAN command line (see Section 2.9)
- Usage recommendations (see Section 2.10)
- Information about bug fixes to the compiler (see Section 2.11)
- Errors found in the documentation produced for VAX FORTRAN V4 (see Section 2.12)

Appendix A describes how to report software errors and contains a sample SPR form.

Associated Documents

- *Guide to VAX/VMS Software Installation*
- *VAX/VMS Install Utility Reference Manual*
- *Guide to VAX/VMS System Management and Daily Operations*
- *VAX/VMS Utility Routines Reference Manual*
- *VAX FORTRAN User's Guide*
- *Programming in VAX FORTRAN*
- *VAX/VMS Symbolic Debugger Reference Manual*

Conventions Used in This Document

Convention	Meaning
UPPERCASE	Uppercase words and letters used in command descriptions indicate that you should type the word or letter exactly as shown.
lowercase	Lowercase words and letters used in command descriptions indicate that you should substitute a word or value of your choice.
SYSGEN> USE CURRENT	Interactive examples show system output (prompt characters and output lines) in black type. All words and values that you enter in response to prompts are printed in red type.

Chapter 1

Installation Guide

You should read this manual and the sections dealing with maintenance updates in the *Guide to VAX/VMS Software Installation before* you begin the installation.

This chapter describes the procedure for installing the VAX FORTRAN compiler (including the FORTRAN system definition library FORSYSDEF.TLB) and the VAX FORTRAN HELP file on the VAX/VMS operating system. The procedure is automated and requires only that you (1) mount the kit volumes when prompted and (2) respond to questions issued during the installation procedure.

After successful installation, you should notify all FORTRAN users that a new version of the compiler has been installed.

This chapter provides information on the following:

- Installation prerequisites (Section 1.1)
- The contents of the software installation package (Section 1.2)
- The procedure for installing the VAX FORTRAN compiler, which builds FORSYSDEF.TLB on site (Section 1.3.1)
- The procedure for installing the VAX FORTRAN HELP file (Section 1.3.2)
- The procedure for installing the VAX FORTRAN compiler as a known shared image (Section 1.4)
- Information on maintenance updates (Section 1.5)

1.1 Installation Prerequisites

The following is a checklist of items that are required to perform an installation:

1. System manager privileges.
2. For a complete installation, a minimum of 6500 blocks of free disk space (approximately 3400 blocks are used after installation).

   The following table gives the approximate disk space requirements for installing individual kit components.

	FREE BLOCKS REQUIRED	BLOCKS USED
FORTRAN COMPILER	1500	1100
FORSYSDEF.TLB	5500	1500
large HELP	2000	600
small HELP	800	100

3. Thirty to ninety minutes of installation time, depending on your type of media and your system configuration.
4. The *Guide to VAX/VMS Software Installation.*
5. VAX/VMS V4.0 or later.
6. Approximately 600 contiguous free global pages and one free global section, if the VAX FORTRAN compiler is to be installed as a known image (see Section 1.4).

1.1.1 Small-Disk System Installation Requirements

Small-disk systems with less than 70,000 blocks of system disk space are referred to as tailored systems. Tailored systems have two additional installation requirements over those of regular, nontailored systems.

1. If you plan to build the FORSYSDEF text library during installation, the file STARLETSD.TLB must exist on your system in SYS$LIBRARY. (STARLETSD.TLB is shipped on the VAX/VMS OPTIONAL saveset.)

If the OPTIONAL saveset was *not* restored during the VAX/VMS upgrade, STARLETSD.TLB must be selectively restored. To obtain this file from the saveset, mount the volume of the VAX/VMS kit containing the OPTIONAL saveset and execute the following backup command:

```
$ BACKUP/SELECT=STARLETSD.TLB dev:[0.0]OPTIONAL/SAV SYS$LIBRARY
```

where 'dev' is the device where the OPTIONAL saveset is mounted.

This file can reside on either the system disk or the library disk. If you want to put STARLETSD.TLB on the library disk in order to save space on your system disk, you must use the VMSTAILOR MOUNT command to mount the library disk for write access prior to restoring the file.

2. The system object libraries, STARLET.OLB and IMAGELIB.OLB, must *not* have been tailored to the system disk prior to the VAX FORTRAN installation. If these libraries are on the system disk, the VMSTAILOR DELETE command must be used to remove them. This requirement will be eliminated in a future release of VAX/VMS.

1.2 Contents of the Software Installation Package

The VAX FORTRAN V4 installation package consists of two separate kits.

1. The first kit contains the VAX FORTRAN compiler and related files. In addition, it contains the files required to build FORSYSDEF.TLB on site. Refer to Appendix C in the *VAX FORTRAN User's Guide* for more information on the contents of FORSYSDEF.
2. The second kit contains the VAX FORTRAN HELP files.

If your installation medium is a tape, either magnetic tape or TK50, both kits are on a single medium. Otherwise, there will be separate media volumes for each kit. Your cover letter specifies the number and contents of your media.

1.3 Installation Procedure

CAUTION

> You should perform the VAX FORTRAN installation procedure when there are no active users on the system. Unlike previous versions of the VAX FORTRAN installation procedure, warnings are given if there are active users when the procedure is run. The compiler kit should install successfully, even with active users on the system; however, the HELP kit will fail if DCL HELP is being accessed.

The installation procedure is in two parts, either of which may be performed first.

- Part 1 — the procedure for installing the VAX FORTRAN compiler and related files, except for VAX FORTRAN HELP, and for building a new FORSYSDEF on site (see Section 1.3.1).
- Part 2 — the procedure for installing VAX FORTRAN HELP (see Section 1.3.2).

You begin installation of VAX FORTRAN V4 on any VAX/VMS V4.0 (or later) system by first logging in to the system manager's account.

Refer to the *Guide to VAX/VMS Software Installation* for complete information on the VMSINSTAL command procedure.

NOTE

> If you install from the console drive, the standard console medium will not be present in the console drive. Therefore, the system may not be able to reboot following a power failure or system failure.
>
> If a system failure occurs during the installation process, the system bootstrap procedure will detect that the installation was interrupted and will restart it. VMSINSTAL will prompt you to resume installation at the appropriate point.

1.3.1 Installing the VAX FORTRAN Compiler and FORSYSDEF.TLB

NOTE

If a previous version of the VAX FORTRAN compiler has been installed as a known image, VMSINSTAL will automatically attempt to install the new version as a known image. Therefore, you must be sure that there are at least 600 free contiguous global pages and 1 free global section available. You can determine whether the compiler has been installed as a known image and you can also check the number of global pages and global sections using the procedures described in Section 1.4.

To install either the compiler or FORSYSDEF.TLB, do the following:

1. Ensure that the prerequisites listed in Section 1.1 have been met.
2. Invoke VMSINSTAL by typing a command of the form:

   ```
   $ @SYS$UPDATE:VMSINSTAL [FORT04n] [device]
   ```

   The command file SYS$UPDATE:VMSINSTAL invokes the VAX/VMS installation procedure.

   The first parameter identifies the product to be installed. For the VAX FORTRAN compiler or FORSYSDEF, the parameter is in the following form: the product name, FORT; the version number, 04; and the number of the update being installed (substitute the update number for n).

   The second parameter identifies the device where the installation media is to be mounted.

   If you fail to enter the product name or device at this point, VMSINSTAL will prompt you for the information later.

   The installation procedure responds with the following messages:

   ```
   VAX/VMS Software Product Installation Procedure

   It is dd-mmm-yyyy at hh:mm.
   Enter a question mark (?) at any time for help.
   ```

3. The messages are followed by a series of questions. You must answer Y (YES), N (NO), or ? for help. The bracketed response is the default.

   ```
   * Are you satisfied with the backup of your system disk [YES]?  yes
   ```

If you have performed the necessary backup, enter YES or press the RETURN key. If you reply with NO, the installation procedure is aborted.

4. The installation procedure now asks you to mount the first volume of the VAX FORTRAN compiler kit on the drive. Type YES when ready.

```
Please mount the first volume of the set on 'device'.
* Are you ready?  yes
%MOUNT-I-MOUNTED, FORT01        mounted on 'device':
```

5. The installation procedure then continues with the following message:

```
The following products will be processed:

  FORT V4.n

       Beginning installation of FORT V4.n at hh:mm

%VMSINSTAL-I-RESTORE, Restoring product saveset A...
```

6. If your VAX FORTRAN compiler kit contains more than one volume, the installation procedure prompts you for the second volume and waits for you to type YES after you have inserted it on the device. If you respond NO to the prompt, the command procedure repeats the prompt.

```
%BACKUP-I-READYREAD, mount volume 2 on 'device': for reading
Enter "YES" when ready:  yes
```

7. The installation procedure now asks if you want to purge the files that were replaced during the installation.

```
* Do you want to purge files replaced by this installation [YES]?  yes
```

If you type YES, all older versions of the compiler and its related files are deleted and replaced with new files; purging is recommended. If you type NO, the older files will remain.

NOTE

> If you have a system that requires tailoring, you may need to purge old versions of the compiler because you may not have sufficient space for the new compiler.

8. The installation procedure asks if you want the compiler to be installed.

```
* Do you wish to have the VAX FORTRAN compiler installed [YES]?  yes
```

If you wish to only update FORSYSDEF.TLB, enter NO. If you enter NO, skip to step 12; if you enter YES, continue with the next step.

9. The installation procedure then continues with the following message:

```
+-----------------------------------------------------------+
|            Installation of VAX FORTRAN V4                 |
|                         Compiler                          |
+-----------------------------------------------------------+
```

10. The installation procedure asks if you want to execute the Installation Verification Procedure (IVP).

```
This kit contains an Installation Verification Procedure
to verify  the correct  installation  of the VAX FORTRAN
compiler.

* Do you want to run the IVP after the installation [YES]? yes
```

11. The installation procedure then asks if you want a printed copy of the edit history file. In response, type the number of copies desired (0 if none).

```
This kit contains a file, VAXFORUPD.MEM, summarizing the
changes made to the VAX FORTRAN  compiler since its last
release.

* How many copies would you like to print [0]: 1
Job n entered on queue 'printer name'
```

    A copy of VAXFORUPD.MEM is always moved to SYS$UPDATE.

12. The installation procedure now prints information concerning the FORSYSDEF library and asks if you want to build a new FORSYSDEF.TLB.

```
A new FORSYSDEF.TLB is available with this installation.

In order to build your FORSYSDEF library, this procedure
requires at least 5500 blocks of  available  disk space,
most of  which  are used for  temporary work files.  The
FORSYSDEF library itself will  take  approximately  1500
blocks of disk space upon completion of  this  procedure
and will be placed in your SYS$LIBRARY area.

                          NOTE
Before installing the kit, be sure to have read Section
1.5.1.1, UPDATING FORSYSDEF.  It addresses the question
of when a new FORSYSDEF.TLB should be built.

* Do you wish to build a new FORSYSDEF.TLB [NO]? yes
```

    If you chose to build a new FORSYSDEF.TLB, the installation procedure responds as follows:

```
Building FORSYSDEF. This should take about 20 minutes
```

13. The installation procedure now lists the file(s) that the installation created or modified. The files updated will vary, depending on whether the compiler and/or FORSYSDEF were installed.

```
Your VMS system will now be updated to include the
following new and modified file(s):

SYS$SYSTEM:FORTRAN.EXE          [new]
SYS$MESSAGE:FORTERR1.EXE        [new]
SYS$MESSAGE:FORTERR2.EXE        [new]
SYS$LIBRARY:FORTV4CLD.CLD       [new]
SYS$TEST:FORTTEST.COM           [new]
SYS$TEST:FORSYSDEFTST.COM       [new]
SYS$LIBRARY:DCLTABLES.EXE       [modified]
SYS$UPDATE:VAXFORUPD.MEM        [new]
SYS$LIBRARY:FORSYSDEF.TLB       [new]

%VMSINSTAL-I-MOVEFILES, files will now be moved to their target directories...
```

14. If you chose to execute the Installation Verification Procedure, the following message is issued:

```
+-------------------------------------------------------+
|           Verification Command Procedure for          |
|                     VAX FORTRAN                       |
+-------------------------------------------------------+
```

The IVP compiles, links, and executes a test program. When the IVP has completed, you should receive the following message:

```
VAX FORTRAN V4.n-'edit number' TEST PASSED
```

where n is the version number of the VAX FORTRAN compiler that you are installing.

15. When the kit has been successfully installed, you receive the following messages, and the installation procedure returns you to DCL level:

```
Installation of FORT V4.n completed at hh:mm

        VMSINSTAL procedure done at hh:mm
$
```

16. If installation was performed on a cluster common system disk, and you will not be rebooting the cluster after completing the installation, an additional step must be performed.

SYS$LIBRARY:DCLTABLES.EXE has been modified during the installation. VMSINSTAL has replaced it as a known shared image on the system that the installation was performed on.

You must invoke the INSTALL Utility on all other systems in the cluster and replace SYS$LIBRARY:DCLTABLES.EXE. The same holds if the compiler is a known shared image on the cluster. You must invoke the INSTALL Utility on all other systems in the cluster and replace SYS$SYSTEM:FORTRAN.EXE. Refer to Section 1.4.3 for information on using the INSTALL Utility's REPLACE command.

1.3.2 Installing the VAX FORTRAN HELP File

CAUTION

If any user attempts to access DCL HELP during the installation of VAX FORTRAN HELP, the installation may fail. If the installation fails, you must restart the installation procedure. VAX FORTRAN HELP files are not corrupted by installation failures.

To install the HELP files for VAX FORTRAN, do the following:

1. Invoke VMSINSTAL by typing a command of the form:

   ```
   $ @SYS$UPDATE:VMSINSTAL [FHLP04n] [device]
   ```

   The command file SYS$UPDATE:VMSINSTAL invokes the VAX/VMS installation procedure.

   The first parameter identifies the product to be installed. For VAX FORTRAN HELP, the parameter is in the following form: the product name, FHLP; the version number, 04; and the number of the update being installed, (substitute the update number for n).

   The second parameter identifies the device where the installation media is to be mounted.

   If you fail to enter the product name or device at this point, VMSINSTAL will prompt you for the information later.

   The procedure responds with the following messages:

   ```
   VAX/VMS Software Product Installation Procedure

   It is dd-mmm-yyyy at hh:mm.
   Enter a question mark (?) at any time for help.
   ```

2. The messages are followed by a series of questions. You must answer Y (YES), N (NO), or ? for help. The bracketed response is the default.

   ```
   * Are you satisfied with the backup of your system disk [YES]? yes
   ```

   If you have performed the necessary backup, enter YES or press the RETURN key. If you reply with NO, the installation procedure is aborted.

3. The installation procedure asks you to mount the first, and only, volume of the HELP kit on the drive. Type YES when ready.

   ```
   Please mount the first volume of the set on 'device'.
   * Are you ready? yes
   %MOUNT-I-MOUNTED, FHLP01        mounted on 'device':
   ```

4. The installation procedure then continues with the following messages:

   ```
   The following products will be processed:

     FHLP V4.n

           Beginning installation of FHLP V4.n at hh:mm

   %VMSINSTAL-I-RESTORE, Restoring product saveset A...

   +-----------------------------------------------------+
   |            Installation of VAX FORTRAN V4            |
   |                      FORTRAN HELP                    |
   +-----------------------------------------------------+
   ```

5. The installation procedure now generates information on the HELP files and prompts you for the version you would like installed. (See Section 2.5 for more information on the two HELP files.)

   ```
   This  kit  contains two separate  HELP  files,  a  large
   version (approximately 600 blocks) including information
   on FORTRAN  language  features,  and  a  smaller version
   (approximately 100 blocks)  describing only the  FORTRAN
   command.  Both versions contain on-line release notes.

   Do you want to install the larger version of FORTRAN HELP [YES]? yes
   ```

   If you type YES, the larger version is installed; if you type NO, the small version is installed.

6. When the HELP installation has completed, the installation procedure will issue the following messages and return you to DCL level:

```
Your VMS system will now be updated to include the
following new and modified file(s):

SYS$HELP:HELPLIB.HLB       [modified]

%VMSINSTAL-I-MOVEFILES, files will now be moved to their target directories...

Installation of FHLP V4.n completed at hh:mm

        VMSINSTAL procedure done at hh:mm
$
```

1.4 Defining VAX FORTRAN as a Known Shared Image with the INSTALL Utility

If the VAX FORTRAN compiler is used frequently at your site, it should be specified as a known shared image to the VAX/VMS system. This designation reduces both the overhead associated with invoking the compiler and the memory requirements when more than one user is compiling FORTRAN programs.

To specify the compiler as a known shared image, use the VAX/VMS INSTALL Utility, which is fully described in the *VAX/VMS Utilities Reference Manual* and briefly in the *Guide to VAX/VMS System Management and Daily Operations.* (The SYSGEN facility is also described in the same manuals.)

If the compiler has been previously installed as a known shared image, VMSINSTAL has already made the new version known. In that case, you do not need to repeat the procedure.

NOTE

You should be logged in under the system manager's account when performing the following operations. Also, you should perform these operations on a system that has just been boot-strapped to ensure that a DELETE specification was not given to the INSTALL Utility. The DELETE specification may cause the available space in the global page table to become fragmented. Because contiguous space is required for known shared images, fragmentation of the available global page table space could cause the INSTALL Utility to fail, even when sufficient total space is available.

The following definition must be in effect in order for the INSTALL commands described in Sections 1.4.1–1.4.3 to work properly.

```
INSTALL :== $SYS$SYSTEM:INSTALL/COMMAND_MODE
```

1.4.1 Preparation

Before specifying VAX FORTRAN as a known shared image, you must ensure that the system SYSGEN parameters GBLPAGES (global pages) and GBLSECTIONS (global sections) are large enough to accommodate it. You do this by using the INSTALL and SYSGEN Utilities as follows:

1. Invoke the VAX/VMS INSTALL Utility by typing:

   ```
   $ INSTALL
   ```

2. Display the existing global sections already known to VAX/VMS by typing:

   ```
   INSTALL> LIST/GLOBAL
   ```

   The INSTALL Utility lists the following:

   - All known global sections
   - The number of global sections used
   - The number of global pages used and unused

   The VAX FORTRAN compiler uses approximately 600 global pages and one global section. If fewer than 600 global pages remain unused, you can increase the number of available global pages by deleting an existing known image or by increasing the GBLPAGES system parameter with the SYSGEN Utility.

   Type EXIT or CTRL/Z to exit the INSTALL Utility.

3. Display the existing number of available global sections by typing:

   ```
   $ RUN SYS$SYSTEM:SYSGEN
   SYSGEN> USE CURRENT
   SYSGEN> SHOW GBLSECTIONS
   ```

   The system responds with global sections data. The current maximum number of global sections is the first number. The FORTRAN kit requires one global section. If all of the global sections are used (compare the number shown by SYSGEN with the number currently used as shown by INSTALL), you must increase the number of available global sections by deleting an existing global section or by increasing the GBLSECTIONS parameter with the SYSGEN Utility.

NOTE

> Neither GBLPAGES nor GBLSECTIONS are dynamic SYSGEN parameters. If you must change either to accommodate the VAX FORTRAN compiler, your system must be rebooted.

4. In addition, if you are installing the VAX FORTRAN compiler with privileges, then all shareable images used must also be installed. These shareable images include the callable interface to the VAX Common Data Dictionary, CDDSHR.EXE, and the two message files, FORTERR1.EXE and FORTERR2.EXE. This is meant as a security feature in VAX/VMS to prevent a user from activating a private (noninstalled) image with privileges. Likewise, if your system already has the callable interface to the VAX Common Data Dictionary, CDDSHR.EXE, installed as a protected image, then the two message files FORTERR1.EXE and FORTERR2.EXE must be installed as well.

1.4.2 Using INSTALL at System Bootstrap Time

It is desirable to have the compiler automatically specified as a known shared image during the system bootstrap procedure (see the *Guide to VAX/VMS System Management and Daily Operations*). You can do this by modifying the site-specific startup command procedure, SYS$MANAGER:SYSTARTUP.COM, as follows:

1. Check that the INSTALL command is contained in the command file. If not, insert the following command:

```
$ INSTALL
```

2. Specify FORTRAN.EXE in the list of image names following the INSTALL Utility invocation. When a command is given, VAX/VMS searches the list of known images in the order opposite to that in which they were installed. Therefore, the compiler should be included after less frequently used images but before more frequently used images. To specify the compiler in this list of image names, insert the command:

```
INSTALL> ADD SYS$SYSTEM:FORTRAN.EXE/SHARED/OPEN/HEADER_RESIDENT
```

The image becomes a known shared image to VAX/VMS the next time the system is bootstrapped.

1.4.3 Using the INSTALL Utility During Timesharing

To specify the VAX FORTRAN compiler as a known shared image on the currently running system without rebooting the system, or to replace an old version of the compiler with a new version, use the following procedure:

1. Ensure that enough global pages and sections are available (see Section 1.4.1.).
2. To determine whether an older version of the VAX FORTRAN compiler is a known image, type:

   ```
   INSTALL> LIST /STRUCTURE
   ```

   The system responds with a list of current known images.
3. If the compiler is already a known image, replace it with the following command:

   ```
   INSTALL> REPLACE SYS$SYSTEM:FORTRAN.EXE
   ```

4. If compiler is not already a known image, specify the location of its image by typing:

   ```
   INSTALL> ADD SYS$SYSTEM:FORTRAN.EXE/SHARED/OPEN/HEADER_RESIDENT
   ```

1.5 Update Releases

DIGITAL may periodically issue update releases of VAX FORTRAN. Each update will consist of a set of new distribution media labeled similarly to the media in the original release. To update your system, use the procedures described in Section 1.3 and the cover letter accompanying the media.

Note that when installing any VAX FORTRAN update releases, you do not need to install previous versions. A VAX FORTRAN update completely replaces any previous versions.

1.5.1 Updates to the Compiler Kit

Each update release of the VAX FORTRAN compiler has the following changes:

- The version number is incremented. For example, if the original compiler had a version number of V4.0-1 and four changes were made to the compiler, the compiler's new version number would be V4.1-5.
- A new version of the compiler executable image file, FORTRAN.EXE, is supplied.
- The VAXFORUPD.MEM file is updated to document all changes made to the compiler since the last release of VAX FORTRAN V4.

1.5.1.1 Updating FORSYSDEF

Unlike previous versions of VAX FORTRAN, the mechanism for building FORSYSDEF for V4 is to process the master description of the modules from STARLETSD.TLB during each installation. STARLETSD.TLB is shipped with all VAX/VMS releases. FORSYSDEF.TLB should be built during the initial VAX FORTRAN installation on a system. It does not need to be installed with succeeding VAX FORTRAN updates, but should be installed whenever a VAX/VMS update is performed on the system. In this way, the FORSYSDEF modules are automatically updated to reflect any changes to the VAX/VMS system definitions in that release of VAX/VMS.

The installation process allows the FORSYSDEF.TLB component of the compiler kit to be installed separately.

If the process used to build FORSYSDEF.TLB is modified, the cover letter will emphasize that FORSYSDEF.TLB should be rebuilt during the VAX FORTRAN update.

1.5.2 Updates to the HELP Kit

The HELP files are modified to correct any technical errors or misleading statements. Also, an updated version of the release notes is contained within each HELP file. After installing the HELP update, you may wish to read the release notes by typing:

```
$ HELP FORTRAN RELEASE_NOTES
```

By omitting the installation of the HELP kit, you should be successful in installing VAX FORTRAN even if there are active users. Access by any user of DCL HELP, either directly or indirectly through another utility, will force installation of the HELP kit to fail. Installation of the HELP kit is not required for the compiler to run properly.

1.6 Edit History File

The VAX FORTRAN V4 installation kit provides an edit history file called VAXFORUPD.MEM. This file lists all the changes made to the compiler since the last release. The following information is provided for each change:

1. The edit-level number
2. The SPR number (if there is one) that prompted the change
3. A short description of the problem
4. A short description of the change

The VAXFORUPD.MEM file is updated each time a VAX FORTRAN V4 kit is built. As part of the installation procedure, this file is copied from the kit volume to the directory [SYSUPD] on the target-system disk and can be printed as often as desired. At your request, the installation procedure prints the file from the target disk.

Chapter 2

Release Notes

This chapter contains information about the following issues:

- Changes associated with VAX FORTRAN V4.0 and the update releases V4.1, V4.2, and V4.3 (see Section 2.1)
- Changes to FORSYSDEF (see Section 2.2)
- Known problems with the VAX FORTRAN V4.3 compiler (see Section 2.3)
- Changes implemented in VAX FORTRAN RTL (see Section 2.4)
- Compatibility of VAX FORTRAN V3 and VAX FORTRAN V4 (see Section 2.5)
- Run-time behavior differences between VAX FORTRAN V3 and VAX FORTRAN V4 (see Section 2.6)
- On-line HELP options (see Section 2.7)
- Compiler performance measurements and suggestions for maximizing performance (see Section 2.8)
- Changes to the FORTRAN command line (see Section 2.9)
- Usage recommendations (see Section 2.10)
- Information about bug fixes to the compiler (see Section 2.11)
- Errors found in the documentation produced for VAX FORTRAN V4 (see Section 2.12)

2.1 VAX FORTRAN V4 Enhancements and Language Extensions

The following sections describe new and changed features related to Versions 4.0—4.3 of VAX FORTRAN.

2.1.1 VAX FORTRAN V4.0 Notes

This section lists new and changed items associated with VAX FORTRAN V4.0.

Several language extensions were implemented in the VAX FORTRAN V4.0 compiler. Of these, by far the most important is a record handling capability (item 1, in the list that follows). Where appropriate, references are made to the chapters and sections in *Programming in VAX FORTRAN* in which the extensions are documented in full.

1. *Records* — New data item that can be composed of multiple fields containing aggregate and scalar data items with any mix of the VAX FORTRAN data types. Records are defined by means of structure declaration blocks (delimited by STRUCTURE and END STRUCTURE statements) and assigned to a memory location by means of the RECORD statement.

   Included in the structure declaration block, along with normal variables and arrays, are mapped common areas (union/map declarations) and substructure declarations (nested structure declarations and RECORD statements). (See Chapter 14 and Sections 6.2.5, 8.13, and 8.15.)
2. */EXTEND_SOURCE qualifier to the FORTRAN command and new switches for /SHOW and /WARNINGS qualifiers*:
   - /EXTEND_SOURCE — Extends the range of FORTRAN source text from columns 1 through 72 to columns 1 through 132.
   - /SHOW=SINGLE — Controls whether unused parameter constant names appear in the cross-reference listing.
   - /SHOW=DICTIONARY — Controls whether the FORTRAN source representation of CDD records referenced by DICTIONARY statements are included in the listing file.
   - /WARNINGS=DECLARATION — Controls whether warning messages are issued when undeclared data items are used in a program (similar to adding an IMPLICIT NONE statement to the program units).

- WARNINGS=GENERAL — Controls whether messages in the I (informational) and W (warning) categories are displayed.

3. *Common Data Dictionary (CDD) support and the DICTIONARY statement* — General description of usage and implications (see Sections 3.4 and 3.5.3).
4. *Compiler output* — Storage map section revised to show structured data items.
5. *VOLATILE statement* — Controls whether optimization operations can affect specified variables, arrays, or common blocks (see Section 8.16).
6. *Stream recordtype file support* — Record Management Services (RMS) now supports Stream, Stream-CR, and Stream-LF recordtypes for the OPEN statement (see Chapter 13).
7. *FAB and NAM blocks* — Allocated in heap storage so that they are accessible the entire time that a file is open.
8. *Zero-valued VFEs in format field descriptors* — Permits formats to be used with zero-width character expressions.
9. *List-directed internal file I/O* — Permitted for internal file READ and WRITE operations (see Chapter 11).
10. *Interactive display of NAMELIST group and values* — Displays the current namelist group name and the name of the variables in response to a question mark (?) or (=?) (see Chapter 11).
11. *End-of-line comments* — Permitted in NAMELIST input data. Comments beginning with an exclamation point (!) may be placed anywhere in the input data where an end-of-line can occur, except within a character value. Comment delimiters are equivalent to an end-of-line (see Section 5.4.3.1).
12. *ERRTST and ERR= improved* — ERRTST is now aware of errors trapped by ERR=, END=, and IOSTAT= (see Chapter 11).
13. *BACKSPACE performance improved* — Performance on disk files has been improved, especially for fixed-length records.
14. *Internal files and "recursion"* — Permits internal READ, WRITE, ENCODE, and DECODE statements to be performed while another internal file operation is already in progress. This is permitted when internal file I/O is performed in an AST routine, exception handler, or a routine specified in an I/O list or variable format expression.
15. *Improved messages* — Additional informational messages are displayed for errors in certain I/O operations.

16. *FOR$RAB provided* — Returns the address of the RMS Record Access Block (RAB) for a given open FORTRAN unit (see the *VAX FORTRAN User's Guide*, Chapter 4).

2.1.2 VAX FORTRAN V4.1 Notes

This section lists new and changed items associated with VAX FORTRAN V4.1.

- *Language-Sensitive Editor support* — The VAX FORTRAN V4.1 kit contains full FORTRAN language support for using the VAX Language-Sensitive Editor. Specific support for the Editor consists of support for the /DIAGNOSTICS qualifier and the long HELP file.
  - */DIAGNOSTICS qualifier* — The /DIAGNOSTICS qualifier specifies that the compiler should produce a diagnostics (.DIA) file describing all errors discovered during compilation. The Language-Sensitive Editor uses this file when the REVIEW command is issued to highlight each error for easy editing.
  - *Language HELP text accessible during editing* — The new language HELP available on the VAX FORTRAN V4 kit has been specifically designed to be accessible during editing in FORTRAN mode in the Language-Sensitive Editor.
- *Release Notes added to HELP* — The VAX FORTRAN V4.0 kit did not contain release notes in the HELP file. Release notes in the HELP file will be updated for each maintenance release of VAX FORTRAN.
- *%FILL now uses LOGICAL*1 arrays instead of CHARACTER strings* — When processing the DICTIONARY statement, the compiler generates source code based on the data structure content specified in the CDD. During this process, it is possible that a data type will be used that is not supported by the FORTRAN language. In this case, the compiler generates a substructure with one unnamed field (%FILL). In V4.0, this field was declared as having a data type of CHARACTER*n, where n was the length of the unnamed field. In V4.1, the LOGICAL*1 data type is used, and the unnamed field is given a dimension of n instead. This allows for unnamed fields of lengths greater than the longest allowed CHARACTER variable.

- *Use of assigned GOTO statements* — In VAX FORTRAN V3, it was possible, but not documented or supported, to pass, as an actual argument to a subprogram, a variable whose value had been assigned a label value in an ASSIGN statement. For instance, the statements

  ```
  ASSIGN 10 TO IVAR
  ...
  CALL SUB(IVAR)
  ```

  show this usage. It was then possible for the subprogram to use this value in an assigned GOTO statement, for example:

  ```
  GOTO IARG
  ```

  This accomplished a "global" control transfer back to the calling program without executing a RETURN statement.

  This method does not work in V4. In V4, because the calling program has no transfers to the label, the label and all associated code are deleted. This means that the assigned GOTO will fail because the target code will not be available.

  To modify your program to allow the use of VAX FORTRAN V4, you must perform the assigned GOTO in the calling program itself. This can be done by returning a status value from the subprogram and performing a conditional branch based on its value. Another possible solution is to use a condition handler in the calling program to signal the status value instead of directly returning it.

2.1.3 VAX FORTRAN V4.2 Notes

This section lists new and changed items associated with VAX FORTRAN V4.2.

- *FORSYSDEF Installation Changes* — The installation of FORSYSDEF.TLB was changed to correct a problem that existed when installing VAX FORTRAN in a directory that had version number limits imposed. Because the installation procedure used workfiles of the same name for each module in FORSYSDEF (that is, the only variation was an increasing version number), the version number limit was frequently exceeded. As a result, FORSYSDEF would contain only the version limit number of modules upon completion of the installation. This problem was corrected in the installation procedure for FORTRAN V4.2.

- *Two new error messages* — The following error messages were added to VAX FORTRAN V4.2:

```
CDDINIVAL       CDD description contains Initial Value attribute
                (ignored).  Informational.
```

  The CDD description being expanded contains a field that specifies an initial value. The initial value has been ignored.

```
CDDALNARY       CDD description specifies an aligned array
                (unsupported).  Informational.
```

  The CDD description being expanded contains an array field whose elements have an individual alignment specified that VAX FORTRAN cannot accommodate. The array has been replaced by a structure of the appropriate size.

2.1.4 VAX FORTRAN V4.3 Notes

The procedure for installing VAX FORTRAN was modified. As a result, the *VAX FORTRAN Installation Guide/Release Notes* was revised and reissued.

2.2 FORSYSDEF Changes

The FORSYSDEF.TLB library has undergone some changes since the V3 FORSYSDEF. Many new modules have been added (see Appendix C of the *VAX FORTRAN User's Guide* for a list of available modules). Some of these modules now contain structure declaration blocks. These new structure declarations have caused a number of modules in the old V3 FORSYSDEF.TLB to undergo some incompatible changes.

The modified modules now include structure declarations instead of EQUIVALENCE declarations, which were not documented and were used infrequently. These modules include ACCDEF, CRDEF, DEVDEV, DMTDEF, IODEF, LADEF, LCKDEF, MNTDEF, MTDEF, OPCDEF, PRVDEF, PSLDEF, RMSDEF, SECDEF, STSDEF, TTDEF, XFDEF, and XMDEF. If a program references any of these names declared in the V3 FORSYSDEF module's EQUIVALENCE statements, it will be necessary to modify the program to use the structure and field names instead. You should notify your FORTRAN users of this change.

If users need to continue accessing the old FORSYSDEF.TLB, the system managers should make a copy of this library and inform their users to use the logical FORT$LIBRARY to access it until all programs are converted to use the new version of FORSYSDEF. In the examples that follow, the old FORSYSDEF.TLB was copied to SYS$LIBRARY:V3FORSYS.TLB. To access this library from your programs, define the FORT$LIBRARY logical name so that the compiler will look for the V3FORSYS.TLB library first:

```
$ DEFINE FORT$LIBRARY SYS$LIBRARY:V3FORSYS.TLB
```

After defining the logical name FORT$LIBRARY, your VAX FORTRAN compiler will first search for the library V3FORSYS.TLB in SYS$LIBRARY: to obtain the required modules. If this library is not found, it will look for the library FORSYSDEF.TLB in SYS$LIBRARY.

Remember to delete this old version after programs depending on it have been converted. The installation procedure will not purge FORSYSDEF.TLB automatically.

NOTE

> The modules $PSMMSG and $SMBMSG in the new FORSYSDEF.TLB are not documented in Appendix C of the *VAX FORTRAN User's Guide*. Both modules contain symbiont messages and can be included by users writing their own symbionts.

2.3 Known Problems with the VAX FORTRAN V4.3 Compiler

There are none at this time.

2.4 VAX FORTRAN RTL Changes

VAX FORTRAN RTL is bundled with VAX/VMS and is therefore independent of VAX FORTRAN releases. The version numbers noted in the following subsections are for VAX/VMS, not VAX FORTRAN.

2.4.1 RTL V4.0 Behavior Changes

The following changes to VAX FORTRAN RTL were implemented in VAX/VMS V4.0:

- The equation used to determine the RMS bucket size (used for RMS multiblock transfers) has been modified. The equation used by VAX/VMS V3 was

  ```
  MIN(((MAX(RECL+24, BLOCKSIZE)+511)/512, 32)
  ```

  For VAX/VMS V4, the equation was modified to be

  ```
  (BLOCKSIZE+511)/512
  ```

  Routines that use BLOCKSIZE and worked properly under VAX/VMS V3 may generate an RMS error, block size not large enough, in VAX/VMS V4.
- The RECL specifier in an INQUIRE statement returns the maximum record length for a file created using the EDT editor with VAX/VMS V4.0 or later — regardless of whether or not the file is open. Previously, if the file was not opened, RECL returned the length of the longest record in the file. This change in the value returned for RECL is due to a change in the behavior of the EDT editor.

2.4.2 RTL V4.1 Fix Descriptions

The following changes to the VAX FORTRAN RTL were implemented in VAX/VMS V4.1:

- Change BACKSPACE operations in order to position segmented files properly when the files contain records larger than 2044 bytes. Before the change, some routines that used BACKSPACE operations were failing at run time.
- Change the behavior of I/O routines used for transmitting an entire implied-DO list when the upper bound is less than the lower bound. Before the change, an incorrect number of values were being transmitted.
- Change the OPEN processing routine to properly check for invalid KEY specifications when opening an existing file. Before the change, no error was generated if the KEY positions specified were inconsistent with the key used when the file was created.

2.4.3 RTL V4.2 Fix and Restriction Descriptions

The following change to the VAX FORTRAN RTL is implemented in VAX/VMS V4.2:

- A spurious CLOSE error, file not found, may be encountered when a FORTRAN routine specifying DISPOSE=DELETE opens a file. This problem resides in the Files–11 XQP and will be corrected by VAX/VMS V4.2.

The following restriction is effective with VAX/VMS V4.2:

- Using a WORD key to access an indexed file on a remote node is not supported.

2.5 Compatibility Issues Involving VAX FORTRAN V3 and V4

The following sections describe the compatibility among source programs that are compiled, linked, and run using different versions of VAX FORTRAN and different versions of VAX/VMS.

2.5.1 Source Compatibility

The source language accepted by the VAX FORTRAN V4.0 compiler is upwardly compatible to the source language accepted by the V3 compiler.

2.5.2 Object Compatibility

The object files produced by the VAX FORTRAN V4 compiler are upwardly compatible to V3 object files. They can be mixed freely in the same executable image.

2.5.3 Image Compatibility

The images created by the linker from VAX FORTRAN V4 object modules are compatible with similar images using V3 modules. Images linked on VAX/VMS V4 systems are *not* downward compatible and will not run on VAX/VMS V3 systems.

2.6 Run-Time Behavior

The new optimizations implemented for VAX FORTRAN V4.0 introduce the small chance that the run-time behavior of some FORTRAN programs may change. The development group has devoted considerable time to designing and testing these new optimizations to ensure that this does not happen. During the testing, however, some subtle dependencies and incorrect usages of the language that caused such behavior changes were encountered. These are listed in Sections 2.6.1 through 2.6.5.

2.6.1 Flow Boolean Order

The VAX FORTRAN language does not allow programs to depend on the order of evaluation of expressions. In particular, the order of evaluation of logical expressions in IF statements may change from release to release, so that programs with such dependencies may work in V3 and not in V4.0. For example, the statement

```
IF (RAN(X) .GT. 0.5 .AND. CONDFLAG) GO TO 100
```

will cause the V3 compiler to emit code to call the RAN function first, and if the result is larger than 0.5, the CONDFLAG will be tested. The V4.0 compiler will emit code in the reverse order, to avoid the routine call overhead if the value of CONDFLAG is .FALSE. Thus, programs with this kind of construct will execute according to the language definition, but they will generate different random numbers. If this change causes a problem, you can resolve it by making the test independent of the order of evaluation. For example, the preceding statement may be changed as follows:

```
Y = RAN(X)
IF (Y .GT. 0.5 .AND. CONDFLAG) GO TO 100
```

Thus, the RAN function is always called and the program works the same in both V3 and V4.0 cases.

2.6.2 Subtle Bounds Errors

Programs that address beyond array or string bounds in an uncontrolled way (for example, an undetected logic error) may work differently for VAX FORTRAN V3 and V4.0. These programs are incorrect and should be run with /CHECK=BOUNDS specified in order to find the out of bounds references. They may appear to work properly when compiled with the V3 compiler because of the particular layout of memory generated by the V3 compiler. The V4.0 compiler attempts to optimize memory usage by keeping variables in registers whenever possible. This feature causes the order and number of variables kept in memory locations to change. This change in memory layout may cause such incorrect programs to fail at run time in several different ways.

2.6.3 DMOD More Accurate

The accuracy of the DMOD function was improved in the VAX/VMS V4 Run-Time Library. In particular, in some cases where 0.0D0 was returned in VAX/VMS V3, a number very close to the first argument is now returned.

2.6.4 Use of VAX/VMS Real-Time Features

Some programs that use features of the VAX/VMS operating system intended for real-time programming, such as Asynchronous System Traps (ASTs), may work differently between VAX FORTRAN V3 and V4.0. In particular, it is possible to use FORTRAN variables and arrays to communicate between the FORTRAN program and the relevant AST service routines. This may cause these variables or arrays to change in the middle of the execution of a FORTRAN program unit. The V4.0 compiler assumes that no such asynchronous change is possible unless the variables or arrays involved have been declared VOLATILE.

This problem will be manifested by the program using the previous value of the variable (perhaps because it is now being held in a register) rather than the new value assigned by the AST service routine. Thus, you should examine any program that uses AST service routines written in FORTRAN for such asynchronous updating, and you should declare any variables or arrays used in this way with the VOLATILE statement.

2.6.5 Unformatted Writes of INTEGER*2 Parameter Constants

The following error in the behavior of unformatted WRITE statements has been corrected. If an INTEGER*2 constant declared in a PARAMETER statement was used in an unformatted WRITE statement in a routine compiled with the /I4 qualifier, VAX FORTRAN V3 incorrectly wrote a longword (4 bytes).

The VAX FORTRAN V4 behavior was changed to correctly write a word (2 bytes).

For example, when the following routine was compiled with the /I4 qualifier,

```
INTEGER*2 INTWRITE, INTREAD, INTERR
PARAMETER (INTWRITE = 12)

OPEN(UNIT=3, FORM='UNFORMATTED',STATUS='NEW')
WRITE(3) INTWRITE,INTWRITE
CLOSE(3)
OPEN(UNIT=3, FORM='UNFORMATTED',STATUS='OLD',DISPOSE='DELETE')

READ(3)  INTREAD, INTERR
PRINT *, INTREAD, INTERR
END
```

VAX FORTRAN V3 generated the incorrect results:

```
12       0
```

Under VAX FORTRAN V4, the results are

```
12      12
```

2.7 HELP Options for VAX FORTRAN V4

During the installation procedure, you are given the choice of selecting either an expanded version or a short version of VAX FORTRAN HELP. The first version refers to a new, expanded VAX FORTRAN HELP; the second file name refers to the previous, short VAX FORTRAN HELP.

VAX FORTRAN HELP is included as a separate kit for V4 in order to simplify installing maintenance updates. Refer to Chapter 1 in this manual for specific instructions.

2.7.1 Long HELP

The new, expanded version of HELP, consists of approximately 600 blocks and combines the traditional HELP file with HELP text for VAX FORTRAN language elements. Subtopics include built-in functions, character sets, data, error messages, format specifications, intrinsic functions, parameters, qualifiers, source format, and statements.

2.7.2 Short HELP

The short version of HELP, consists of approximately 100 blocks and contains information on the VAX FORTRAN command line and the command qualifiers. For your convenience, this traditional HELP file has been reformatted for increased readability.

2.8 Compiler Performance and Recommended Working Set

The extensive additional optimizations performed by the VAX FORTRAN V4 compiler, as compared to the V3 compiler, have significantly changed the performance characteristics of the compiler itself.

- Raw compile speeds have degraded marginally.
- Virtual memory usage has approximately tripled.
- The size of the compiler has increased by about 28 percent, from 422 pages to 540 pages.

These changes significantly affect the page faulting behavior of the FORTRAN command when operating in a limited working set environment. Thus, for acceptable compile performance, you should familiarize yourself with the information given in the next sections and adjust accordingly the FORTRAN compilation working sets used by the accounts on your system.

2.8.1 CPU Usage

Raw compile speed, as measured by CPU time spent compiling a given FORTRAN program, has degraded somewhat from that of VAX FORTRAN V3. This degradation is not uniform — some programs suffer more compile speed degradation than others, and some even compile faster. Measurements have shown a range of 10-22 percent degradation for a wide selection of large FORTRAN programs. The degradation is caused by the additional analysis needed to achieve the improved optimizations of the VAX FORTRAN V4 compiler.

2.8.2 Memory Usage

The VAX FORTRAN V4 compiler uses more memory than the V3 compiler in three ways. First, its internal data structures are about twice as large as those in the V3 compiler. Second, it makes about twice as many passes over these data structures. Third, the size of the compiler code has increased by about 28 percent.

All of these changes are critical to the global optimization capability of the VAX FORTRAN V4 compiler. They will also cause the number of page faults encountered by the V4 compiler to increase in comparison to the V3 compiler. Thus, to ensure that the V4 compiler does not place an unacceptable burden on system performance, make sure that an adequate working set is used when compiling FORTRAN programs.

One important difference between the two versions is that the V4 compiler performs analysis on entire program units (routines) at once. That is, for optimization purposes, it simultaneously represents all of the operations in memory for an entire program unit. The V3 compiler, on the other hand, performed optimizations only on bounded "blocks" of statements, and so had a fixed upper limit for the amount of memory needed for optimization purposes, regardless of the size of the program.

The Figures 2-1 and 2-2 graphically show how the number of page faults has increased from VAX FORTRAN V3 to VAX FORTRAN V4.

Figure 2-1: Page Faults Versus Working Set, VAX FORTRAN V3

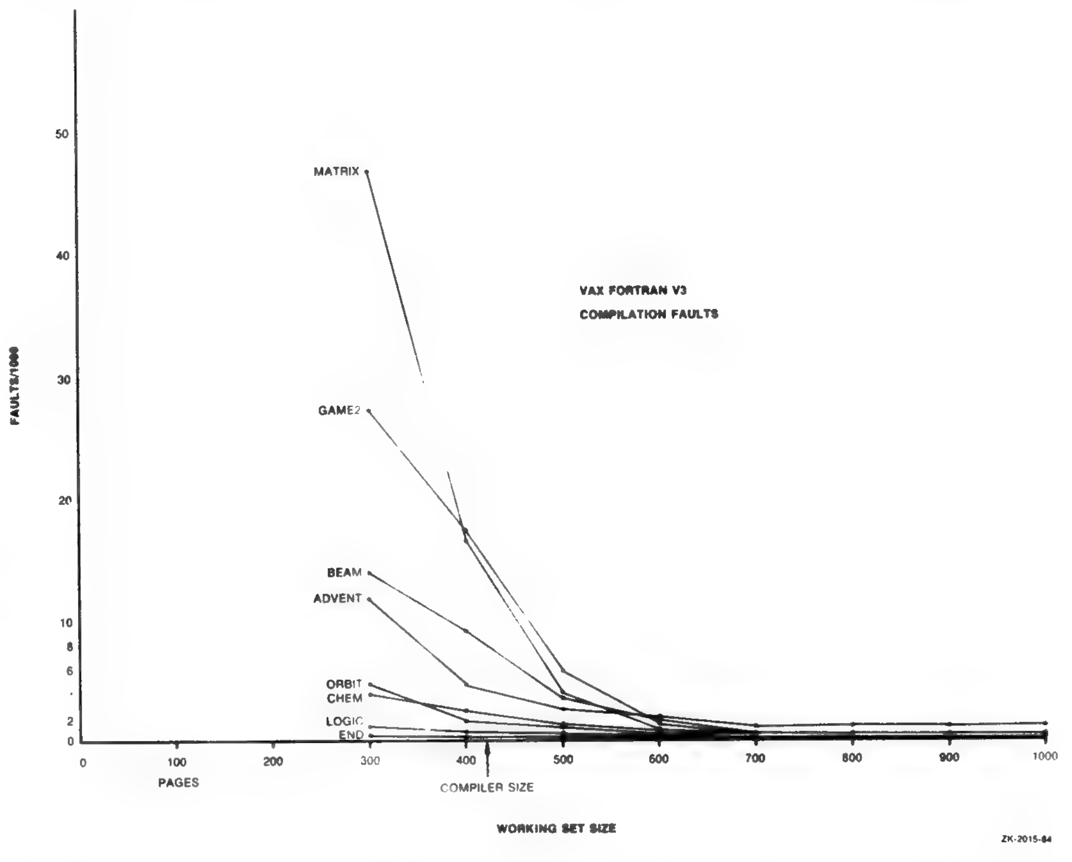


Figure 2-2: Page Faults Versus Working Set, VAX FORTRAN V4

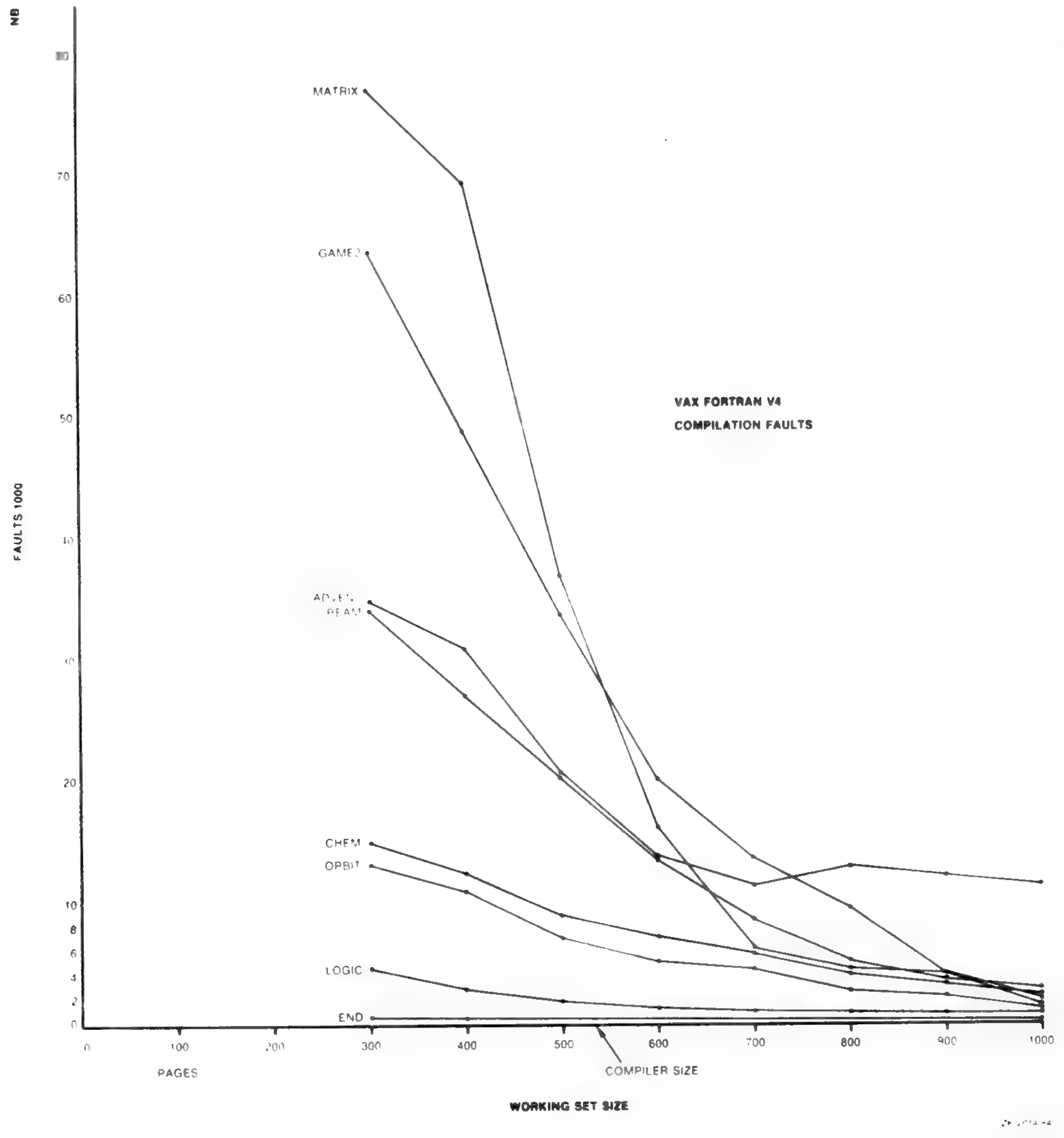


Table 2–1 gives brief descriptions of the programs analyzed in Figures 2–1 and 2–2.

Table 2–1: Program Descriptions for Figures 2-1 and 2-2

Program	Description
END	Single END statement.
LOGIC	1150 lines, 2 routines, no comments, LOGICAL operations only. 2/3 simple assignments, 1/3 expression assignments, no flow at all.
ORBIT	1048 lines, 14 routines, no comments, mixture of statements, calculates satellite orbits.
CHEM	1419 lines, 9 routines, no comments, large scientific program with lots of computations. Chemical engineering.
ADVENT	ADVENTURE computer game, 3092 lines, 1856 lines of source. One very large main program, 2086 lines, 1284 lines of source, a few subroutines.
GAME2	Computer game, large fraction of included text. GAME2.FOR is 5313 lines, 4222 source, GAME2.INC is 104 lines, 72 source lines, included 52 times. This totals 5408 lines of included text, 3744 source lines included for a total compiled of 10721 lines, 7966 source lines.
BEAM	Particle beam simulation, lots of COMPLEX, 3164 lines, 2827 source lines in 35 routines, good size mixture.
MATRIX	Package for analysis and correlation of two dimensional data arrays. 147 routines, most small. 13,154 total lines, 7153 source lines.

The implications of this data and recommendations for working set sizes are given in the text that follows.

The compiler fault rates can be divided into three regimes, based on the relationship between compiler size (number of pages) and the working set size of the process doing the compile. The data in the preceding figures can be summarized, in terms of these regimes, as follows:

- *Large working sets, two or more times the size of the compiler* — In this regime, faults are not excessive for either compiler. The increased data structure size and extra passes dominate in this regime and result in total faults of about three times greater than those of the V3 compiler, regardless of program size or organization. This rate, however, is modest and does not degrade system performance.

- *Middle-size working sets, between one and two times the size of the compiler* — In this regime, the effects of the larger structures and extra passes on fault rates are even stronger. In addition, the extra faulting of the compiler itself begins to be felt in the lower end of the range.

  In this regime, program size and organization begin to make a significant difference. For very small compilations, below approximately 1000 lines of actual code (excluding comments and blank lines), the fault rate is still relatively low for both compilers, even though the fault rate of the V4 compiler is about three times greater than that of the V3 compiler.

  Programs with many small routines degrade much more quickly than programs with a few large routines. This is partly due to the more effective optimization when compiling a large amount of code all at once, but for the most part it is caused by compiler page faults.

  In the case of the small routine, the compiler pages are faulted repeatedly as each routine gets compiled. In the large routine case, only a small part of the compiler is needed in memory at any one time and this allows more data structure pages to be resident.

  For large compilations in this regime, the V4 compiler faults considerably more than the V3 compiler, as much as six times more. This amount of faulting can significantly degrade system performance. For this reason, compiling with the working set at the high end of this range is recommended, especially when large compilations (several thousand code lines) of many small program units are used.
- *Small working sets, less than the size of the compiler* — In this regime, fault rates quickly become excessive for both V3 and V4 compilers. Again, the V4 compiler faults two to three times more than the V3 compiler, but both begin to cause system performance degradation when compiling programs with more than 1000 code lines, especially when many routines are being compiled. Small programs continue to give acceptable fault rates for both compilers in this regime, mostly because many of the compiler pages are never actually needed and because only minimal-size data structures are needed.

In general, for large programs, the working set sizes for the various regimes are as follows:

- The low faulting regime is not attained until about 1000 pages working set for the V4 compiler, versus about 700 pages for the V3 compiler.
- The intermediate faulting regime is between 600 and 1000 pages working set for the V4 compiler, versus 400 to 700 for the V3 compiler.
- The high faulting regime is below 600 pages working set for the V4 compiler, versus below 400 for the V3 compiler.

The working set size differences for VAX FORTRAN V4 versus VAX FORTRAN V3 are based on compiler size differences. The size of the V3 compiler code and static data is 422 pages, and for V4, 540 pages. This is about a 28 percent increase. The boundaries for the regimes reflect this difference.

The preceding observations lead to the following specific recommendations for use of the VAX FORTRAN V4 compiler:

1. Follow the directions for optionally installing the compiler as a known image with the /OPEN, /SHARED, and /HEADER_RESIDENT options. This will keep the compiler startup overhead low and will alleviate system performance degradation when several compilations are in progress simultaneously.
2. If large programs (more than 1000 noncomment lines) are being compiled, ensure that working set sizes are not in the high faulting regime to avoid system performance degradation. Specifically, compiling large programs consisting of many small routines in working set sizes below 600 is to be avoided. Remember that many modest size INCLUDE files often cause compilation size to be much larger than source file size.
3. In systems where large programs are being developed and where memory is very scarce, performance can be improved by encouraging users to break up large programs into small routines in separate source files. In this way, the size of each compilation can be kept small and system performance will not suffer, even in small working sets. Used in conjunction with object library manipulation capabilities of the linker and librarian, this can be an effective way to improve system performance and individual productivity.

2.9 Output File Specifications on Command Lines

The compiler's behavior in determining the directory used for the output object and listing files has been changed to agree with the documentation and other DCL commands. Both the old and new behavior are described in the sections that follow. The recommended changes to command files that depend on the old, incorrect, behavior are also described.

In the following examples, assume that the default device and directory is set to:

```
MYDEV:[MYDIR]
```

2.9.1 Behavior of VAX FORTRAN V3

In VAX FORTRAN V3, and all prior versions, the command:

```
$ FORTRAN [XYZ]MYFILE/OBJECT
```

produced an object file in directory [MYDIR], the default, rather than in directory [XYZ], which contained the input file. This was contrary to the documentation for the specification of output files in the *VAX/VMS DCL Dictionary* and inconsistent with other system language processors, such as the MACRO assembler and the LINK command. The same was true of the /LIST qualifier.

Note that this applies only to the use of the /LIST and /OBJECT qualifiers following an input file specification. When these qualifiers are used following the FORTRAN command, the use of the default directory for the output file is correct and has not changed.

In addition, when a file specification is given explicitly with the qualifier, the normal defaulting rules apply, including use of the default device and directory, even if they are different from the device and directory of the input file. For example, if the command:

```
$ FORTRAN [XYZ]MYFILE/OBJECT=[]
```

were given, the object file is placed in the default directory as given in the object file specification. This behavior is also correct and has not changed.

2.9.2 Behavior of VAX FORTRAN V4

In VAX FORTRAN V4, the command:

```
$ FORTRAN [XYZ]MYFILE/OBJECT
```

produces an object file in directory [XYZ], the same as the input file specification instead of the default directory. This behavior corresponds with the documentation for the behavior of the qualifiers and with the behavior of other system language processors. When the /LIST and /OBJECT qualifiers are used after the FORTRAN command and an explicit file specification is given with the qualifier is correct, the behavior has not changed.

2.9.3 How to Change Command Files

If you have command files that use the form of the FORTRAN command described previously, you will need to modify them to accommodate this change. You need to supply the default file specification explicitly with the output qualifier. Thus, to modify the command:

```
$ FORTRAN [MYDIR.FORTRAN.SOURCE]PROG1 FOR/OBJECT/LIST
```

so that the output files are placed in the default directory [MYDIR], you need to add explicit default file specifications to the qualifiers. For example:

```
$ FORTRAN [MYDIR.FORTRAN.SOURCE]PROG1.FOR/OBJECT=[]/LIST=[]
```

will work in the same way as the previous command with both VAX FORTRAN V3 and V4.

2.10 Usage Recommendations

Users may want to modify existing programs to conform to the programming conventions described in the following list. (Note: The third item is a mandatory change.)

- Use of %LOC — Many programs obtain the values for system symbols by declaring the symbols as EXTERNAL and referencing the value as %LOC(symbol). This in now unnecessary in most cases. Instead, you should INCLUDE the relevant module from FORSYSDEF.TLB (see the *VAX FORTRAN User's Guide*, Appendix C for a list of modules) and refer to the symbol without the %LOC.
- See HELP information under V3_to_V4 for a discussion on the use of the COMMON and EQUIVALENCE declarations found in some V3 FORSYSDEF modules.
- When (1) using shared variables and arrays in handlers and (2) capturing the address of subprogram arguments, you should use the VOLATILE statement (see the *VAX FORTRAN User's Guide*, Section 1.3.2.2).
- Use of USEROPEN routines — Using the new RECORD manipulation capability of VAX FORTRAN V4, it is now possible to write USEROPEN routines in FORTRAN instead of MACRO. See Chapter 4 of the *VAX FORTRAN User's Guide* for a general discussion of using RMS from FORTRAN programs. Section 4.2.1 contains examples of FORTRAN USEROPEN routines.

2.11 Information about Bug Fixes

A list of bugs and fixes is contained in the file VAXFORUPD.MEM, shipped with each VAX FORTRAN kit. This file can be loaded and printed by the installer as part of the automated installation procedure. If you have any programs that have been modified or compiled with nonstandard options because of a compiler bug, return them to their original form and recompile normally when you receive a new update. If the bug persists, please submit an SPR describing the problem so that it can be corrected before the next update.

2.12 Documentation Errors

This section lists the documentation changes for the September 1984 versions of *Programming in VAX FORTRAN* (Order No. AA-D034D-TE), *VAX FORTRAN User's Guide* (Order No. AA-D035D-TE), and *VAX FORTRAN Language Summary* (Order No. AV-M763B-TE). It also lists errors discovered in the on-line HELP documentation.

Note that asterisks (*) are placed in the front of the page number references for those documentation errors that were found since the V4.2 release of VAX FORTRAN. All of the other errors were documented previously in the on-line release notes issued with the V4.0—V4.2 releases of VAX FORTRAN.

2.12.1 Changes to On-Line HELP Documentation

Descriptions of BTEST, IBCLR, IBITS, and IBSET intrinsic functions — The low-order bit position for these functions is bit position 0, not 1. The HELP file has been modified accordingly.

2.12.2 Changes to *Programming in VAX FORTRAN*

The following changes will be reflected in the next update or revision of *Programming in VAX FORTRAN*:

- General

  All discussions concerning relative organization files, occurring principally in Chapters 11 and 13, should appear in blue ink. Relative organization is an extension to the ANSI standard.

- Page 2-9

  The EDT operations associated with the keys in the VT200 keypad are the same as those for the VT100 keypad. The rightmost block of keys in Figure 2-3 should be the same as the lower block of keys (starting with PF1/GOLD) in Figure 2-2.

- Page 3-5

  The heading for the third column in Table 3-1 should be "Default," not "Qualifier."

- *Page 3-13

  The description of the SINGLE option should read as follows:

  "[SINGLE] specifies that symbolic names of parameter constants are to be included in cross-reference listings — even if they are not referenced outside the PARAMETER statements in which they are declared. The negative form, NOSINGLE, specifies that names of parameter constants are to be suppressed if they are only declared and not referenced elsewhere. This is useful for cross-reference listings of small programs that specify INCLUDE declarations but use only a small number of the parameter constant names declared."

- *Page 3-26

  The form of the OPTIONS statement shown on this page is incorrect. Qualifiers specified in an OPTIONS statement are not separated by commas.

- Page 3-27

  In the first paragraph in Section 3.5.3, the slash (/) in "STRUCTURE /END STRUCTURE" should be an ellipsis (...), not a slash. The ellipsis in this case is a symbolic representation of any necessary statements in the body of a structure declaration.

- Page 6-31
  - The first example at the top of the page should be as follows:

    ```
    NEXT_APP.APP_MEMO(1)(1:1)
    ```

  - In Section 6.2.6, the definition of "scalar reference" should be as follows:

    "Scalar Reference — resolves itself to a reference to a single, typed data item (a variable, scalar record field, array element, constant, or character substring) or an expression."

- Page 6-32

  In the "array name references" category in the list of example references, "RECARY" should be in blue ink.

- Page 8-4

  In the description of the "nlist" symbolic abbreviation, the words "character substring names" should be in blue ink; it is nonstandard to use a character substring as a "nlist" item.

- Page 8-5

  In the description of the "dlist" symbolic abbreviation, the words "character substring names" should be in blue ink; it is nonstandard to use a character substring as a "dlist" item.

- Page 8-10

  The description of the "nlist" symbolic abbreviation should state that a dummy argument cannot be specified in an EQUIVALENCE statement.

- Page 8-25

  In Section 8.15, the description of the STRUCTURE statement (item 1 in the bulleted list) should read as follows:

  "This statement indicates the beginning of a structure declaration and defines the name of the structure."

- *Page 9-21

  In the list of commands at the end of Section 9.8, the DEBUG command should appear in blue ink.

- Page 11-4

  The first column in Table 11-1 should be labeled "Type of Access," not "Statement Category."

- *Page 11-32

  In the paragraph under the heading "Prompting for Current Values," the last sentence should read as follows:

  "If you precede the question mark with an equal sign (that is, =?), the group name and current values of the namelist entities for that group will be displayed as in namelist output (see Section 11.5.1.3)."

- Page 12-3

  The fourth paragraph in Section 12.2 should refer to 12.2.3, not 12.2.1.

- Page 12-21

  The third line in Table 12-2 should be as follows:

  ```
  I,O,Z              INTEGER*4,LOGICAL*4        12
  ```

- Page 13-1

  In Section 13-1, the keyword specification should be enclosed in braces, indicating that one of the two forms must be selected:

  ```
  {keywd      }
  {keywd=value}
  ```

- Page 13-2

  The fourth line item within the first element of the bulleted list should read as follows: "STATUS or TYPE — file existence status at OPEN." Also, "or TYPE" should appear in blue ink, and the phrase "file existence status at OPEN" should appear in black ink, not blue.

- Page 13-3

  In Table 13-1, the value 'PRINT/DELETE' associated with the keywords DISPOSE and DISP should not have an interior apostrophe as shown in the manual.

- Page 13-7

  The last paragraph in Section 13.1.4 should state that the maximum size of RMS multiblock transfers is 63 512-byte blocks, not 32.

- Page 13-19

  A comma is missing from the "file" form of the INQUIRE statement format. The comma belongs after the first square bracket, that is, before the DEFAULTFILE keyword.

- *Page 13-21

  The following sentence should be appended to the last paragraph in Section 13.3.6:

  "Ex is also assigned the value .FALSE. if the file exists but cannot be opened."

- Page 13-30

  In the illustration of the format of the UNLOCK statement, "unit" is a keyword and should be capitalized:

  ```
  UNLOCK ([UNIT=]u[,ERR=s][,IOSTAT=ios])
  ```

- Page 17-9

  The two arguments shown in the example in Section 17.5.1.3 should be preceded by %REF.

- Page 17-21

  The first sentence on this page should refer to SET BREAK and SET TRACE, not SET BREAK and SET WATCH.

- *Page A-6

  In the table of valid and invalid octal integer constants in Section A.5, the third item in the "valid" column ("17777") should appear in the "invalid" column.

- Page A-7

  In the example of a main program, all periods shown in the example statements should be commas. This error also occurs in the first line of the subprogram example. In addition, the subprogram example contains the following errors:

  - Spurious right parenthesis in line 2.
  - Typographical error in second FUNCTION statement.
  - SINDEG assignment statement should appear as follows:

  ```
  SINDEG = SIN(X*3.1459/180)
  ```

- Page B-2

  In Table B-1, the equal sign (=) in row 2, column 2 should be a double quotation mark (") and the pyramid in row 3, column 2 should be a pound sign (#).

- Page E-5

  The mnemonic of the error message second from the bottom should be BADRECREF, not BADRECFREF.

- Page E-8

  The descriptions of the error messages EXTMIXCOM and EXTMIXEQV are interchanged. Also, the message that belongs with EXTMIXEQV should be amended to read as follows:

  "A numeric variable or numeric array element cannot be equivalenced to a character variable or character array element."

- Page E-22

  The description of the MULDECNAM error message should be as follows:

  "A name appears in two or more inconsistent declaration statements or a dummy argument is specified in an EQUIVALENCE statement."

- Page E-26

  In the description of the UNSUPPTYPE error message, "CHARACTER %FILL of the appropriate length" should say "LOGICAL*1 %FILL of the appropriate dimension". For additional information, see HELP information provided by RELEASE_NOTES/V4_to_V4.1.

- INDEX

  Underscores are omitted from references to D_floating, F_floating, G_floating, and H_floating data and from references to various FORTRAN command qualifiers (/D_LINES, G_FLOATING, and others). Also, throughout the Index, indention for second and third level entries is applied inconsistently and may cause some confusion.

- INDEX-14

  The first index entry under "H" should be as follows:

  ```
  H field descriptor, 12-20
  ```

2.12.3 Changes to *VAX FORTRAN User's Guide*

The following changes will be reflected in the next update or revision of the *VAX FORTRAN User's Guide*:

- General

  References to the RTL manual are incorrect; the correct name of the manual is *VAX/VMS Run-Time Library Routines Reference Manual*.

- Page 1-16

  The examples shown under the first bulleted item are incorrectly aligned. They should appear as follows:

Unoptimized Code	**Optimized Code**
I = 100	I = 100
...	...
IF (I .LT. 1) GO TO 100	A(100) = 3.0*Q
A(I) = 3.0*Q	

- Page 1-38

  Bullet 8 should say "X(I-1,J)".

- Page 2-2

  Table 2-1 is misaligned. It should appear as follows:

PSECT Name	Use	Attributes
$CODE	Executable code	PIC,CON,REL,LCL,SHR,EXE,RD, NOWRT,LONG
$PDATA	Read-only data: literals, read-only FORMAT statements	PIC,CON,REL,LCL,SHR,NOEXE, RD,NOWRT,LONG
$LOCAL	Read/write data local to the program unit: user local variables, compiler temporary variables, argument lists, and descriptors	PIC,CON,REL,LCL,NOSHR,NOEXE, RD,WRT,LONG
$BLANK	Blank common block	PIC,OVR,REL,GBL,SHR,NOEXE, RD,WRT,LONG
name(s)	Named common block(s)	PIC,OVR,REL,GBL,SHR,NOEXE, RD,WRT,LONG

- Page 4-14 and 4-15

  The name of several FORSYSDEF modules are incorrect: $XAB should be $XABDEF; $XABDAT should be $XABDATDEF; and $XABPRO should be $XABPRODEF.

- Page 4-23

  Spurious dollar signs are contained in the two RECORD statements shown in the example of a USEROPEN procedure.

- Page 4-25

  In Table 4-1, the equation used to calculate the RMS bucket size (FAB$B_BKS) is no longer correct as shown and should be changed to (BLOCKSIZE+511)/512

- *Page 6-6

  In the list of notes on Page 6-6, the last element in the list should refer to SIGARGS(n+1), not SIGARGS(+1).

- Page C-4 and C-5

  Two modules in the new FORSYSDEF.TLB are not listed. The modules omitted are $PSMMSG and $SMBMSG. Both modules contain symbiont messages and can be included by users writing their own symbionts.

- Page D-7 through D-8

  Not all footnote flags in the example text correctly correspond to the footnotes themselves. Also, some footnote flags are missing. This will be corrected in the next update or revision.

- Page D-8

  The options file used to link the two programs and the illustration of the sample use of the two source programs was omitted. The following material should precede "Notes" on Page D-8:

  The options file PAGEFIL.OPT contains the following line of source text:

```
PSECT_ATTR=MYCOM,PAGE                               (1)
```

  Sample use:

```
$ FORTRAN PAGEFIL1
$ FORTRAN PAGEFIL2
$ LINK PAGEFIL1,PAGEFIL/OPTIONS                     (1)
$ LINK PAGEFIL2,PAGEFIL/OPTIONS                     (1)

$ RUN PAGEFIL1                              ****Process 1****
Waiting for PAGEFIL2 to update section
Modified data in the global section:
     2    4    6    8   10   12   14   16   18   20
    22   24   26   28   30   32   34   36   38   40
    42   44   46   48   50   52   54   56   58   60
    62   64   66   68   70   72   74   76   78   80
    82   84   86   88   90   92   94   96   98  100
$

$ RUN PAGEFIL2                              ****Process 2****
Original data in the global section:
     1    2    3    4    5    6    7    8    9   10
    11   12   13   14   15   16   17   18   19   20
    21   22   23   24   25   26   27   28   29   30
    31   32   33   34   35   36   37   38   39   40
    41   42   43   44   45   46   47   48   49   50
$
```

- Page D-12

  The second comment in the example source program on this page should refer to "row 2, column 15," not "row 5, column 15."

2.12.4 Changes to *VAX FORTRAN Language Summary*

The following changes will be reflected in the next update or revision of the *VAX FORTRAN Language Summary*:

- Page 1

  In the description of the order in which text libraries are searched, the second item should refer to FORT$LIBRARY (not FOR$LIBRARY) in both instances and the third item should refer to SYS$LIBRARY (not FOR$LIBRARY).

Appendix A

Software Performance Reports

Occasionally, you may encounter problems or errors when using the VAX FORTRAN software and documentation. These problems should be communicated to Digital Equipment Corporation by means of a Software Performance Report (SPR) form, such as the one shown in Figure A-1. These forms may be obtained from the nearest SPR center.

You can submit Software Performance Reports to the nearest SPR center for handling. SPRs are forwarded to the appropriate group within the Software Engineering Department for analysis and response.

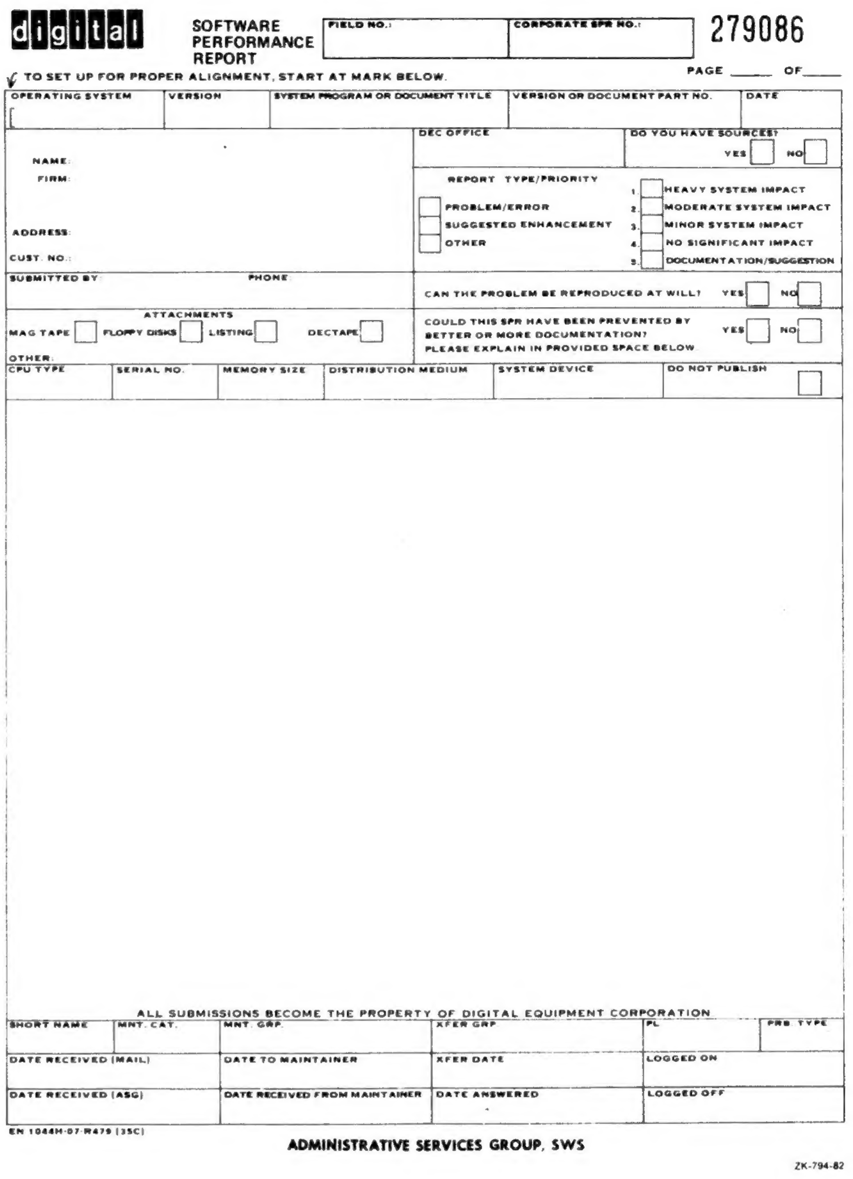
In preparing a Software Performance Report, you should adhere to the following guidelines:

1. Give as complete a description as possible of the problem encountered. Often a detail that may seem irrelevant will give a clue to solving the problem.
2. If possible, isolate the problem to a small example. Large, unfamiliar programs are difficult to work with and may result in a misunderstanding of what the problem is or an inability to duplicate the problem.
3. If the error example is longer than one page of source code, try to send all information in a machine-readable form. Problems in this form are much easier to diagnose. All media are returned.
4. Send console samples, command files, listings, link maps, include files, data files, and so on with the SPR. Annotations showing where the error occurred are extremely helpful.
5. If a program reads input data, include sample input listings, and, if possible, sample output.

6. If an error example cannot be isolated to a single program unit, include listings and other relevant material on all program units involved.

Experience shows that many SPRs do not contain sufficient information to duplicate or identify the problem. Complete and concise information will help DIGITAL give accurate and timely service to software problems.

Figure A–1: Software Performance Report (SPR) Form



digital SOFTWARE PERFORMANCE REPORT

FIELD NO.:	CORPORATE SPR NO.:

279086

PAGE ____ OF____

✓ TO SET UP FOR PROPER ALIGNMENT, START AT MARK BELOW.

OPERATING SYSTEM	VERSION	SYSTEM PROGRAM OR DOCUMENT TITLE	VERSION OR DOCUMENT PART NO.	DATE

NAME:
FIRM:
ADDRESS:
CUST. NO.:

DEC OFFICE

DO YOU HAVE SOURCES? YES ☐ NO ☐

REPORT TYPE/PRIORITY

☐ PROBLEM/ERROR
☐ SUGGESTED ENHANCEMENT
☐ OTHER

1. ☐ HEAVY SYSTEM IMPACT
2. ☐ MODERATE SYSTEM IMPACT
3. ☐ MINOR SYSTEM IMPACT
4. ☐ NO SIGNIFICANT IMPACT
5. ☐ DOCUMENTATION/SUGGESTION

SUBMITTED BY: PHONE:

CAN THE PROBLEM BE REPRODUCED AT WILL? YES ☐ NO ☐

ATTACHMENTS

MAG TAPE ☐ FLOPPY DISKS ☐ LISTING ☐ DECTAPE ☐

OTHER:

COULD THIS SPR HAVE BEEN PREVENTED BY BETTER OR MORE DOCUMENTATION? YES ☐ NO ☐
PLEASE EXPLAIN IN PROVIDED SPACE BELOW

CPU TYPE	SERIAL NO.	MEMORY SIZE	DISTRIBUTION MEDIUM	SYSTEM DEVICE	DO NOT PUBLISH
					☐

ALL SUBMISSIONS BECOME THE PROPERTY OF DIGITAL EQUIPMENT CORPORATION

SHORT NAME	MNT. CAT.	MNT. GRP.	XFER GRP.	PL / PRB. TYPE
DATE RECEIVED (MAIL)		DATE TO MAINTAINER	XFER DATE	LOGGED ON
DATE RECEIVED (ASG)		DATE RECEIVED FROM MAINTAINER	DATE ANSWERED	LOGGED OFF

EN 1044H-07-R479 (35C)

ADMINISTRATIVE SERVICES GROUP, SWS

ZK-794-82

VAX FORTRAN
Installation Guide/Release Notes
AA–H953E–TE

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

____

____

____

____

____

____

Did you find errors in this manual? If so, specify the error and the page number.

____

____

____

____

____

____

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) ______________________

Name ______________________ Date ____________

Organization ______________________________

Street ______________________________

City ______________________ State ____________ Zip Code or Country ________

Do Not Tear - Fold Here and Tape

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line